# Liver Disease Classification using Machine Learning EDX Data Science - Capstone Project

Muditha Hapudeniya

2/12/2022

## Contents

---

## Introduction

The liver is one of the major organs in the human body. It performs many essential biological functions and detoxification necessary for digestion and growth. The liver is located in the right upper quadrant of the abdomen below the diaphragm. The liver produces bile which breakdown fat. The liver's highly specialized tissue, consisting of primarily hepatocytes, regulates a wide variety of biochemical reactions. When the liver cells are damaged, some of the chemicals such as bilirubin, AGOT, and SGPT produced by the liver can be detected in the blood.

Signs and symptoms of liver damage do not appear until the final stages of the disease. Therefore it is vital to identify the liver damage early and start the treatment early. Biochemical changes of liver damage can be detected in blood in the early stages. These tests are called liver function tests which measure blood levels of Total bilirubin, Alanine transaminase (ALT), Aspartate transaminase (AST), AST/ALT ratio, Alkaline phosphatase (ALP), Gamma-glutamyltransferase (GGT) and Albumin.

Machine learning techniques can be used to identify patients who has a risk of developing liver disease and provide appropriate treatment earlier. The UCL Machine Learning Repository has published a data set on Indian Liver patients for researchers to test their machine learning models to accurately classify the liver disease patients (1).

This data set contains information on 583 Indian liver patients; out of them, 416 are liver patient records, and 167 are non-liver patient records. The data set was collected from the northeast of Andhra Pradesh, India.

The data set contains the following information.

1. Age of the patient
2. Gender of the patient
3. Total Bilirubin
4. Direct Bilirubin
5. Alkphos Alkaline Phosphatase
6. Alamine Aminotransferase
7. Aspartate Aminotransferase
8. Total Proteins
9. Albumin
10. Albumin and Globulin Ratio
11. Presence of the disease

Since this is a publicly available data set, many researchers have used the data set to learn machine learning and test their models. I have found research articles where people have applied models such as Naïve Bayesian classifier, Support Vector Machines (SVM), K Nearest Neighbor (KNN) and artificial neural networks (ANN) and random forest methods. This data set is also popular in Kaggle. Most people have used Python language to build models rather than R. Since I am from a medical background, this data set made me interested in applying machine learning methods to classify the data set.

### Evaluating the models

There are many metrics that can be used to measure the performance of a classifier or predictor. Here I have a binary classification problem, i.e. to classify the patients in two groups - with disease and without the condition. Most commonly, researchers have used measures such as Accuracy, Sensitivity, Precision and Specificity to evaluate the model.

Accuracy: The accuracy of a classifier is the percentage of the test set records that are correctly classified by the classifier.

Sensitivity: Sensitivity is also referred to as True positive rate, i.e. the proportion of positive records that are correctly identified.

Precision: precision is defined as the proportion of the true positives against all the positive results (both true positives and false positives)

Specificity: Specificity is the True negative rate that is the proportion of negative records that are correctly identified

In this report, I will try to apply leaner regression, knn, random forest, loess, and ensemble models to predict the presence of liver daises using the Indian liver disease data set.

## Method

I have downloaded the data set from The UCL Machine Learning Repository and cleaned it to remove rows with missing values. Missing data can interfere with the models and may result in errors or wrong results. Once the cleaning was done, data explorations were done to get familiar with the data set and identify further required data preparation.

## Analysis and results

First of all, I have loaded the required libraries for the analysis.

```r
library(readr)
library(tidyverse)
library(psych)
library(ggplot2)
library(patchwork)
library(dplyr)
library(caret)
library(reshape2)
library(gridExtra)
library(ggcorrplot)
```

Next, I have downloaded the liver disease data set from the UCL Machine Learning Repository website.

```
# Download and load the data set

URL <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian%20Liver%20Patient%20Data
liver <- read_csv(URL, col_names = FALSE)
```

This data set does not contain the column names attached to it. Therefore, I have named the columns as follows.

```
# name the data columns
colnames(liver) <- c("Age", "Gender", "totalBilirubin", "directBilirubin", "totalProteins", "Albumin"
                     ,"AGratio", "SGPT", "SGOT", "Alkphos", "Disease")
```

Now let's look at the structure of the data set

```
# view the structure of the data set
str(liver)
```

```
## spec_tbl_df [583 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Age            : num [1:583] 65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender         : chr [1:583] "Female" "Male" "Male" "Male" ...
##  $ totalBilirubin : num [1:583] 0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7 ...
##  $ directBilirubin: num [1:583] 0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2 ...
##  $ totalProteins  : num [1:583] 187 699 490 182 195 208 154 202 202 290 ...
##  $ Albumin        : num [1:583] 16 64 60 14 27 19 16 14 22 53 ...
##  $ AGratio        : num [1:583] 18 100 68 20 59 14 12 11 19 58 ...
##  $ SGPT           : num [1:583] 6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ...
##  $ SGOT           : num [1:583] 3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.4 ...
##  $ Alkphos        : num [1:583] 0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ...
##  $ Disease        : num [1:583] 1 1 1 1 1 1 1 1 2 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double(),
##   ..   X2 = col_character(),
##   ..   X3 = col_double(),
##   ..   X4 = col_double(),
##   ..   X5 = col_double(),
##   ..   X6 = col_double(),
##   ..   X7 = col_double(),
##   ..   X8 = col_double(),
##   ..   X9 = col_double(),
##   ..   X10 = col_double(),
##   ..   X11 = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Here we could see that all the columns are numerical (double) except the gender column, which is character.

Then I examined the basic statistics for the data set variables. Here, I have used the describe() method of the psych package.

```
# describe the data set variables.
describe(liver)
```

```
##                  vars   n   mean     sd median trimmed   mad  min    max  range
## Age                 1 583  44.75  16.19  45.00   44.84 17.79  4.0   90.0   86.0
## Gender*             2 583   1.76   0.43   2.00    1.82  0.00  1.0    2.0    1.0
## totalBilirubin      3 583   3.30   6.21   1.00    1.74  0.44  0.4   75.0   74.6
## directBilirubin     4 583   1.49   2.81   0.30    0.74  0.30  0.1   19.7   19.6
## totalProteins       5 583 290.58 242.94 208.00  238.40 74.13 63.0 2110.0 2047.0
## Albumin             6 583  80.71 182.62  35.00   43.91 22.24 10.0 2000.0 1990.0
## AGratio             7 583 109.91 288.92  42.00   56.79 31.13 10.0 4929.0 4919.0
## SGPT                8 583   6.48   1.09   6.60    6.51  1.04  2.7    9.6    6.9
## SGOT                9 583   3.14   0.80   3.10    3.15  0.89  0.9    5.5    4.6
## Alkphos            10 579   0.95   0.32   0.93    0.93  0.25  0.3    2.8    2.5
## Disease            11 583   1.29   0.45   1.00    1.23  0.00  1.0    2.0    1.0
##                   skew kurtosis    se
## Age              -0.03    -0.57  0.67
## Gender*          -1.19    -0.58  0.02
## totalBilirubin    4.88    36.70  0.26
## directBilirubin   3.20    11.20  0.12
## totalProteins     3.75    17.52 10.06
## Albumin           6.52    49.95  7.56
## AGratio          10.49   149.10 11.97
## SGPT             -0.28     0.21  0.04
## SGOT             -0.04    -0.40  0.03
## Alkphos           0.99     3.22  0.01
## Disease           0.94    -1.11  0.02
```

According to the basic statistics generated by the psych package, we can see that the mean age of the patient in the sample is 44.75 years. Total proteins, albumin and has a very wide range and skewed.

### Data cleaning

It is important to clean the data set before any analysis. Let's see whether there are any missing values and, if yes, remove them. Those missing values can affect the analysis adversely.

```
# check for missing values
liver[rowSums(is.na(liver))!= 0,]
```

```
## # A tibble: 4 x 11
##     Age Gender totalBilirubin directBilirubin totalProteins Albumin AGratio
##   <dbl> <chr>           <dbl>           <dbl>         <dbl>   <dbl>   <dbl>
## 1    45 Female            0.9             0.3           189      23      33
## 2    51 Male              0.8             0.2           230      24      46
## 3    35 Female            0.6             0.2           180      12      15
## 4    27 Male              1.3             0.6           106      25      54
## # ... with 4 more variables: SGPT <dbl>, SGOT <dbl>, Alkphos <dbl>,
## #   Disease <dbl>
```

```
# remove column which has missing values
liver <- na.omit(liver)
```

There were four rows with missing values. I have removed them from the data set.

### Data preperation

Some of the algorithms may not work well with the character data. Therefore, I have converted the Gender variable to numeric as follows.

```r
# convert gender to numeric factor
liver$Gender[liver$Gender=='Male']<- 1
liver$Gender[liver$Gender=='Female']<- 2
liver$Gender <- as.factor(liver$Gender)
```

In the data set, disease states was marked as 1 for presence and 2 for not presence. I have converted the variable to binary by replacing the value 2 with 0.

```r
# convert predictor variable to binary
liver$Disease[liver$Disease==2]<- 0
liver$Disease <- as.factor(liver$Disease)
```

### Data exploration

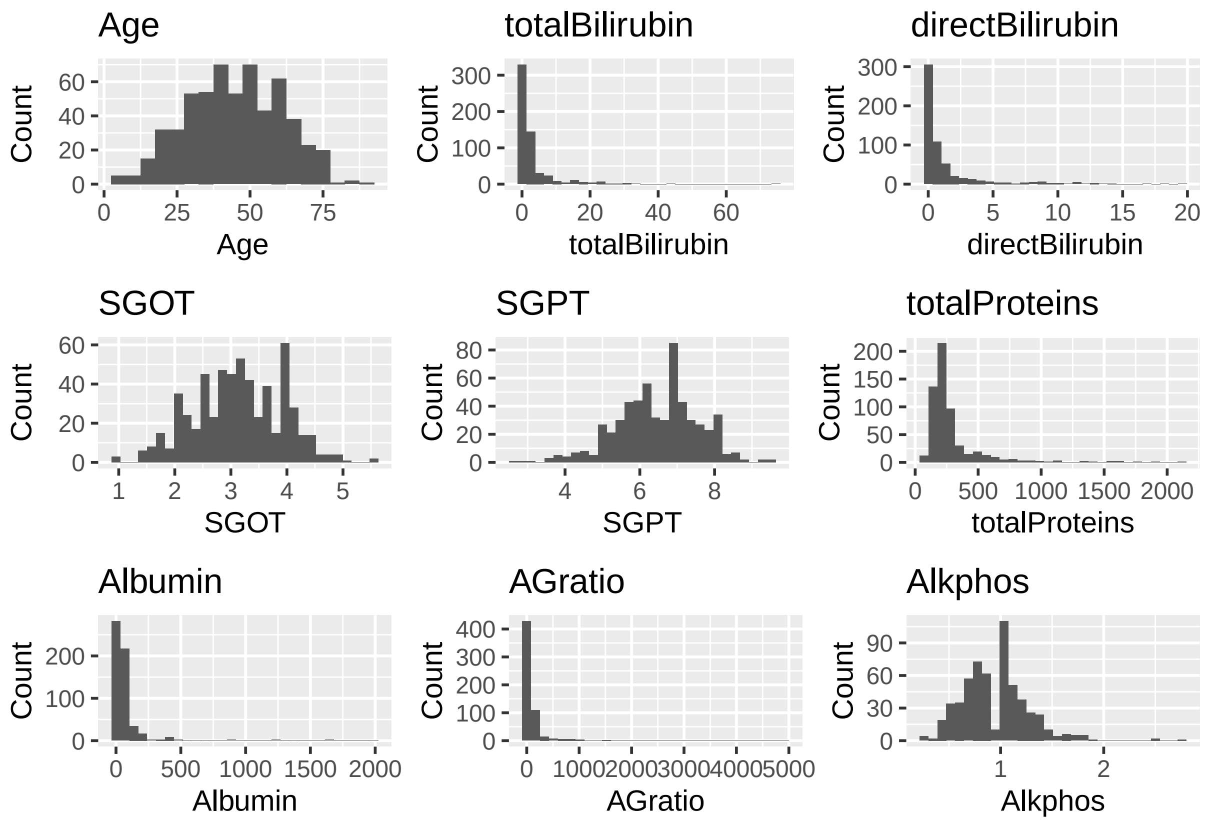
Then I tried to visualize the distribution of the numerical variable.

```r
ph1 <- liver %>% ggplot(aes(x=Age)) +
    geom_histogram(binwidth=5)+
  labs(title = "Age", y = "Count")


ph2 <- liver %>% ggplot(aes(x=totalBilirubin)) +
    geom_histogram()+
  labs(title = "totalBilirubin", y = "Count")


ph3 <- liver %>% ggplot(aes(x=directBilirubin)) +
    geom_histogram()+
  labs(title = "directBilirubin", y = "Count")


ph4 <- liver %>% ggplot(aes(x=SGOT)) +
    geom_histogram()+
  labs(title = "SGOT", y = "Count")


ph5 <- liver %>% ggplot(aes(x=SGPT)) +
    geom_histogram()+
  labs(title = "SGPT", y = "Count")


ph6 <- liver %>% ggplot(aes(x=totalProteins)) +
    geom_histogram()+
  labs(title = "totalProteins", y = "Count")


ph7 <- liver %>% ggplot(aes(x=Albumin)) +
    geom_histogram()+
  labs(title = "Albumin", y = "Count")
```

```r
ph8 <- liver %>% ggplot(aes(x=AGratio)) +
    geom_histogram()+
  labs(title = "AGratio", y = "Count")

ph9 <- liver %>% ggplot(aes(x=Alkphos)) +
    geom_histogram()+
  labs(title = "Alkphos", y = "Count")
```

I arranged the charts as follows for easy visualizing.

```r
(ph1|ph2|ph3)/(ph4|ph5|ph6)/(ph7|ph8|ph9)
```

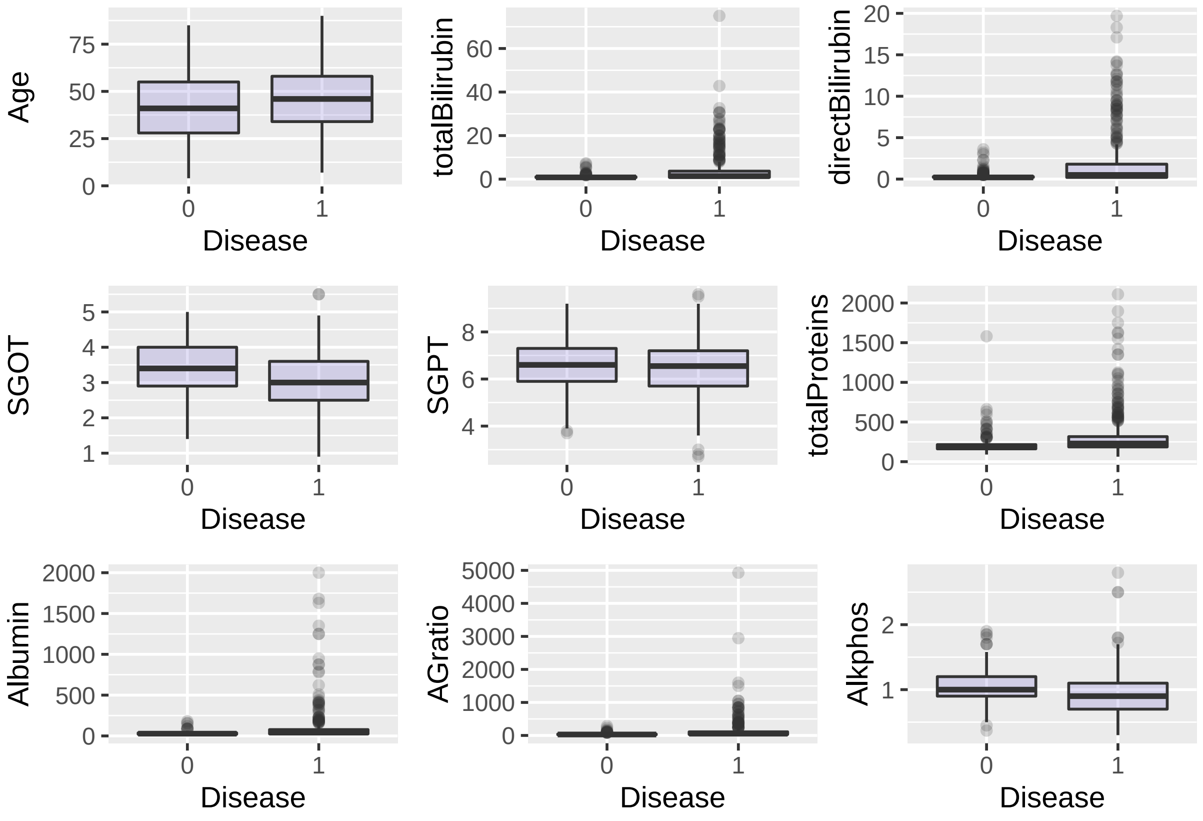


Age, SGOT, SGPT and Alkaline phosphatase are almost normally distributed. However, other variables are not normally distributed and very skewed.

Next, I looked at how the values were distributed in columns for each disease status using box plots.

```r
pb1 <- liver %>% ggplot(aes(x= as.factor(Disease), y=Age)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb2 <- liver %>% ggplot(aes(x= as.factor(Disease), y=totalBilirubin)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb3 <- liver %>% ggplot(aes(x= as.factor(Disease), y=directBilirubin)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")
```

```r
pb4 <- liver %>% ggplot(aes(x= as.factor(Disease), y=SGOT)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb5 <- liver %>% ggplot(aes(x= as.factor(Disease), y=SGPT)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb6 <- liver %>% ggplot(aes(x= as.factor(Disease), y=totalProteins)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb7 <- liver %>% ggplot(aes(x= as.factor(Disease), y=Albumin)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb8 <- liver %>% ggplot(aes(x= as.factor(Disease), y=AGratio)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")

pb9 <- liver %>% ggplot(aes(x= as.factor(Disease), y=Alkphos)) +
    geom_boxplot(fill="slateblue", alpha=0.2) +
    xlab("Disease")


(pb1|pb2|pb3)/(pb4|pb5|pb6)/(pb7|pb8|pb9)
```



The following graphs show the change of values by disease status.

### Predictor variable analysis

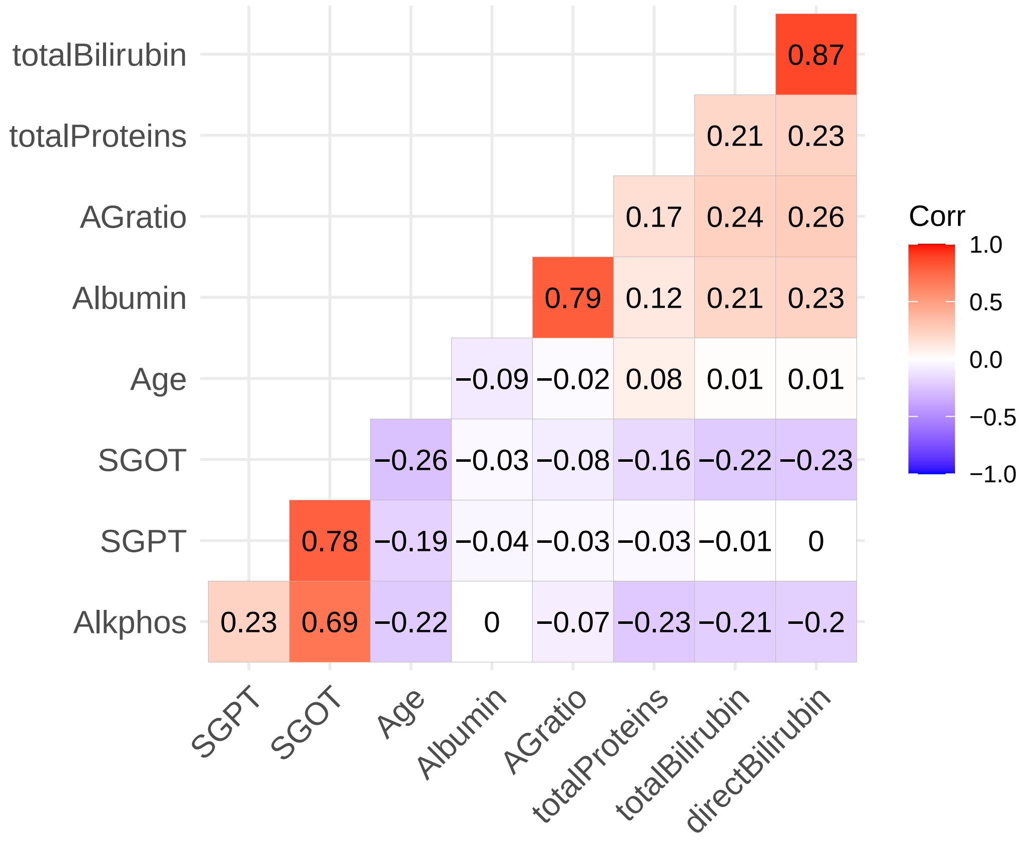
Here I tried to identify the relationship between each variable using a correlation matrix.

```
# calculate the correlations of variables without Gender and Disease Status
correlations <- cor(liver[,c(-2,-11)])
correlations
```

```
##                          Age totalBilirubin directBilirubin totalProteins
## Age              1.000000000    0.011000374    6.784303e-03    0.07887835
## totalBilirubin   0.011000374    1.000000000    8.744810e-01    0.20573917
## directBilirubin  0.006784303    0.874480969    1.000000e+00    0.23400757
## totalProteins    0.078878350    0.205739173    2.340076e-01    1.00000000
## Albumin         -0.087799162    0.213375493    2.331801e-01    0.12477671
## AGratio         -0.020498946    0.237323055    2.570224e-01    0.16657999
## SGPT            -0.186248122   -0.007905923    3.270877e-05   -0.02706202
## SGOT            -0.264210935   -0.222086570   -2.284092e-01   -0.16341865
## Alkphos         -0.216408346   -0.206267186   -2.001247e-01   -0.23416650
##                      Albumin     AGratio          SGPT        SGOT     Alkphos
## Age              -0.08779916 -0.02049895 -1.862481e-01 -0.26421094 -0.21640835
## totalBilirubin    0.21337549  0.23732305 -7.905923e-03 -0.22208657 -0.20626719
## directBilirubin   0.23318008  0.25702239  3.270877e-05 -0.22840915 -0.20012469
## totalProteins     0.12477671  0.16657999 -2.706202e-02 -0.16341865 -0.23416650
## Albumin           1.00000000  0.79186215 -4.243210e-02 -0.02865750 -0.00237499
## AGratio           0.79186215  1.00000000 -2.575101e-02 -0.08491457 -0.07003983
## SGPT          -0.04243210 -0.02575101  1.000000e+00  0.78311217  0.23488718
## SGOT          -0.02865750 -0.08491457  7.831122e-01  1.00000000  0.68963234
## Alkphos       -0.00237499 -0.07003983  2.348872e-01  0.68963234  1.00000000
```

Let's visualize it with a correlation plot.

```
ggcorrplot(correlations, hc.order = TRUE, type = "lower",  lab = TRUE)
```



**Removing the highly correlated variables**

The highly correlated variable can drive the analysis to one side and can give biased results. Therefore, to get the best performance, we need to remove these highly correlated variables from the analysis.

```
# find highly correlated variables
high_coor_cols <- findCorrelation(correlations, cutoff = 0.7, names = TRUE)
high_coor_cols
```

```
## [1] "SGOT"            "directBilirubin" "AGratio"
```

```
# remove highly correlated data
liver_data <- liver[, !names(liver) %in% high_coor_cols]
dim(liver_data)
```

```
## [1] 579   8
```

Here, we can see that the SGOT, directBilirubin and AGratio are highly correlated. I have removed those variables from the data set.

### Dividing the data set into training and test data set

I have divided the data set into training and testing sets. 80% of the data were taken as a training set and 20% for the testing.

```
# set a seed
set.seed(2021)
# split data set into train and test
index <- createDataPartition(liver_data$Disease, p = 0.8, list = FALSE)
train = liver_data[index,]
test = liver_data[-index,]
dim(train)
```

```
## [1] 464   8
```

```
dim(test)
```

```
## [1] 115   8
```

I got 464 patients for training and 115 for testing.

### Logistic regression

```
# Logistic Regression - GLM
# set seed
set.seed(2021)
# fit/train the model
model1_glm_fit <- train(Disease ~., data = train, method = "glm", family = "binomial")

# predict outcomes on test
model1_glm_pred <- predict(model1_glm_fit, test)
# confusion matrix
model1_cm <- confusionMatrix(model1_glm_pred, test$Disease)
model1_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  7  5
##          1 26 77
##
##                Accuracy : 0.7304
##                  95% CI : (0.6397, 0.8089)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.383747
##
##                   Kappa : 0.1866
##
##  Mcnemar's Test P-Value : 0.000328
##
##             Sensitivity : 0.21212
##             Specificity : 0.93902
##          Pos Pred Value : 0.58333
##          Neg Pred Value : 0.74757
##              Prevalence : 0.28696
##          Detection Rate : 0.06087
##    Detection Prevalence : 0.10435
##       Balanced Accuracy : 0.57557
##
##        'Positive' Class : 0
##
```

```
# model accuracy
model1_acc <- model1_cm$overall["Accuracy"]
model1_sen <- model1_cm$byClass["Sensitivity"]
model1_spe <- model1_cm$byClass["Specificity"]
model1_pre <- model1_cm$byClass["Precision"]
model1_acc
```

```
##  Accuracy
## 0.7304348
```

**knn classification**

```
# knn
# set seed
set.seed(2021)
# fit/train the model
model2_knn_fit <- train(Disease ~ ., data = train, method = "knn")

# predict outcomes on test
model2_knn_pred <- predict(model2_knn_fit, test)
# confusion matrix
model2_cm <- confusionMatrix(model2_knn_pred, test$Disease)
model2_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  6  8
##          1 27 74
##
##                Accuracy : 0.6957
##                  95% CI : (0.6029, 0.778)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.700515
##
##                   Kappa : 0.1018
##
##  Mcnemar's Test P-Value : 0.002346
##
##             Sensitivity : 0.18182
##             Specificity : 0.90244
##          Pos Pred Value : 0.42857
##          Neg Pred Value : 0.73267
##              Prevalence : 0.28696
##          Detection Rate : 0.05217
##    Detection Prevalence : 0.12174
##       Balanced Accuracy : 0.54213
##
```

```
##        'Positive' Class : 0
##
```

```
# model accuracy
model2_acc <- model2_cm$overall["Accuracy"]
model2_sen <- model2_cm$byClass["Sensitivity"]
model2_spe <- model2_cm$byClass["Specificity"]
model2_pre <- model2_cm$byClass["Precision"]
model2_acc
```

```
##  Accuracy
## 0.6956522
```

### Random forest

```
# Random Forest
# set seed
set.seed(2021)
# fit/train the model
model3_rf = train(Disease ~ ., method = "rf", data = train, prox = TRUE)

# predict outcomes on test
model3_pred <- predict(model3_rf, test)
# confusion matrix
model3_cm <- confusionMatrix(model3_pred, test$Disease)
model3_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  8 11
##          1 25 71
##
##                Accuracy : 0.687
##                  95% CI : (0.5938, 0.7702)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.76681
##
##                   Kappa : 0.124
##
##  Mcnemar's Test P-Value : 0.03026
##
##             Sensitivity : 0.24242
##             Specificity : 0.86585
##          Pos Pred Value : 0.42105
##          Neg Pred Value : 0.73958
##              Prevalence : 0.28696
##          Detection Rate : 0.06957
##    Detection Prevalence : 0.16522
##       Balanced Accuracy : 0.55414
##
##        'Positive' Class : 0
##
```

```r
# model accuracy
model3_acc <- model3_cm$overall["Accuracy"]
model3_sen <- model3_cm$byClass["Sensitivity"]
model3_spe <- model3_cm$byClass["Specificity"]
model3_pre <- model3_cm$byClass["Precision"]
model3_acc
```

```
##  Accuracy
## 0.6869565
```

### Naive Bayes classification

```r
# Naive Bayes
# set seed
set.seed(2021)
# fit/train the model
model4_nb_fit <- train(Disease ~ ., data = train, method = "nb")


# predict outcomes on test
model4_nb_pred <- predict(model4_nb_fit, test)
# confusion matrix
model4_cm <- confusionMatrix(model4_nb_pred, test$Disease)
model4_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23 26
##          1 10 56
##
##                Accuracy : 0.687
##                  95% CI : (0.5938, 0.7702)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.76681
##
##                   Kappa : 0.3318
##
##  Mcnemar's Test P-Value : 0.01242
##
##             Sensitivity : 0.6970
##             Specificity : 0.6829
##          Pos Pred Value : 0.4694
##          Neg Pred Value : 0.8485
##              Prevalence : 0.2870
##          Detection Rate : 0.2000
##    Detection Prevalence : 0.4261
##       Balanced Accuracy : 0.6899
##
##        'Positive' Class : 0
##
```

```r
# model accuracy
model4_acc <- model4_cm$overall["Accuracy"]
model4_sen <- model4_cm$byClass["Sensitivity"]
model4_spe <- model4_cm$byClass["Specificity"]
model4_pre <- model4_cm$byClass["Precision"]
model4_acc
```

```
##  Accuracy
## 0.6869565
```

### Generalized Additive Model using LOESS

```r
# gamLoess
# set seed
set.seed(2021)
# fit/train the model
model5_nb_fit <- train(Disease ~ ., data = train, method = "gamLoess")
#model5_nb_fit

# predict outcomes on test
model5_nb_pred <- predict(model5_nb_fit, test)
# confusion matrix
model5_cm <- confusionMatrix(model5_nb_pred, test$Disease)
model5_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  8  8
##          1 25 74
##
##                Accuracy : 0.713
##                  95% CI : (0.6212, 0.7935)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.546781
##
##                   Kappa : 0.1712
##
##  Mcnemar's Test P-Value : 0.005349
##
##             Sensitivity : 0.24242
##             Specificity : 0.90244
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.74747
##              Prevalence : 0.28696
##          Detection Rate : 0.06957
##    Detection Prevalence : 0.13913
##       Balanced Accuracy : 0.57243
##
##        'Positive' Class : 0
##
```

```r
# model accuracy
model5_acc <- model5_cm$overall["Accuracy"]
model5_sen <- model5_cm$byClass["Sensitivity"]
model5_spe <- model5_cm$byClass["Specificity"]
model5_pre <- model5_cm$byClass["Precision"]
model5_acc
```

```
##  Accuracy
## 0.7130435
```

Now let's see the accuracies got from each machine learning algorithm.

```r
# Overview accuracy
# create table with accuracies overview
tibble(method = c("glm", "knn", "random forest","naive bayes","gamLoess"),
       Accuracy = c(model1_acc, model2_acc, model3_acc, model4_acc, model5_acc),
       Sensitivity = c(model1_sen, model2_sen, model3_sen, model4_sen, model5_sen),
       Specificity = c(model1_spe, model2_spe, model3_spe, model4_spe, model5_spe),
       Precision = c(model1_pre, model2_pre, model3_pre, model4_pre, model5_pre))
```

```
## # A tibble: 5 x 5
##   method        Accuracy Sensitivity Specificity Precision
##   <chr>            <dbl>       <dbl>       <dbl>     <dbl>
## 1 glm              0.730       0.212       0.939     0.583
## 2 knn              0.696       0.182       0.902     0.429
## 3 random forest    0.687       0.242       0.866     0.421
## 4 naive bayes      0.687       0.697       0.683     0.469
## 5 gamLoess         0.713       0.242       0.902     0.5
```

The generalized linear regression model gave the best accuracy out of the five classification models.

### Ensemble

```r
# Ensembler
# list of models
models <- c("glm", "naive_bayes",  "svmLinear", "gamLoess",
            "knn","rf", "avNNet", "nnet" )

# set seed
set.seed(2021)
# fit/train models
fits <- lapply(models, function(model){
  # print(model)
  train(Disease ~ ., method = model, data = train)
})

# predict with each model
fits_predicts <- sapply(fits, function(fits){
  predict(fits, test)
})

# calculate accuracies for each model
acc <- colMeans(fits_predicts == test$Disease)
#acc
```

```r
# obtain mean score of predictions
votes <- rowMeans(fits_predicts == 1)
# accumulated voting, below mean of 0.5 vote is 0
y_hat <- ifelse(votes > 0.5, 1, 0)
# ensembler accuracy mean is mean of accumulated votes accuracy
ensembler_mean <- mean(y_hat == test$Disease)


# which individual models are better than the ensembler mean
better <- ensembler_mean < acc
```

Mean of ensemble evaluation

```r
ensembler_mean
```

```
## [1] 0.7217391
```

```r
acc[which(better==T)]
```

```
## [1] 0.7304348
```

## Discussion and conlusion

This report contains the performance evaluation of several machine learning algorithms using the Indian Liver Patient data set and the cleaning, pre-processing, visualizations, and feature selection required to build better performing models. This project aimed to determine which algorithm among logistic regression, knn, loess, random forest, Naive Bayes and ensemble gives the best prediction accuracy. I got the highest accuracy for the generalized linear regression model, which was 0.73. However, the sensitivity is low. All the models seem to have high specificity but low sensitivity. The ensemble method uses several learning algorithms and gives the mean value. It did not show a higher precision than glm method. We can see that 71.5% of patients have disease in the data set. Even if a physician guess the patient has the condition every time, he is accurate 71.5% of the time. The glm algorithm performed very slightly better than guessing.

We can see that the data set is not balanced as there is a very high number of liver patients than healthy patients. In reality, there are more healthy people than people with liver disease. The high specificity could be due to the unbalanced nature of the data set. I think the models are not suitable in real-life situations due to the unbalanced data set.

However, the data set is an excellent resource for learning machine learning for people like me. Overall, this analysis has offered me many new insights into machine learning, particularly into classification predictions. My knowledge is still too limited and narrow to fully grasp all the possibilities and paths I could have taken in the analysis.

In conclusion, I would like to say a big thank you to our teacher as well as other supporting teachers for opening up my eyes to this vast expanse of the new and exciting domain. My next step is to continue learning and return to this report and try to make it better one day.

## References

1. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.